Learning How to Learn is Learning with Point Sets

Thomas P. Minka
Vision and Modeling Group
MIT Media Laboratory
20 Ames Street
Cambridge, MA 02139

Rosalind W. Picard
Vision and Modeling Group
MIT Media Laboratory
20 Ames Street
Cambridge, MA 02139

Abstract

It has been proposed that learning how to learn be understood in terms of "learning a prior," from a Bayesian point-of-view (Baxter, 1996b). This paper presents an alternative interpretation: learning how to learn is ordinary learning but on point sets, rather than points. The idea behind learning how to learn is to partition the data, learn a model for the partitions, and then apply this model to new partitions. Ordinary learning methods do the same thing but with individual data points as the partitions. The partitioning for learning how to learn may be recovered automatically and may be applied recursively, leading to "task clustering" models. Virtually all existing approaches fit naturally into this unifying framework, including learning a distance metric and learning internal representations.

1 INTRODUCTION

In the classical learning scenario, we have some independent and identically distributed (IID) samples of a function and want to complete the function, i.e. guess the values elsewhere. A generalization of this is that we have multidimensional points, not necessarily defining a function, and want to complete a new, partially observed point (Bregler and Omohundro, 1994). Furthermore, some points in the training set may themselves be partially observed (missing data) (Ghahramani and Jordan, 1994).

Now consider generalizing all of this to the case where instead of IID points, we have IID sets of points. That is, instead of a training set of points and a testing point, we now have a training set of sets of points and a testing set of points. Like the elements of a single point, the points within these sets need not be IID and they need not be governed by the same probability law across different sets (otherwise, there would be little point in grouping them into sets). However, the sets themselves are IID, meaning the parameters of the probability law that defines each set have been sampled from some underlying distribution. For example, a single sinewave is a set of (x, y) points. A set of sinewaves might be training data for a learner whose job is to complete a partial testing sinewave. The training data can suggest the expected range of frequencies, for example. Another example of a set is a timeseries, where points are generally not IID, but rather Markov.

The goal of this paper is to show how conventional ideas carry over into this generalization and that the result corresponds exactly to "learning how to learn" (Baxter, 1996b) (Thrun, 1995) (Caruana, 1994) (Omohundro, 1995).

2 THEORY

The conventional Bayesian learning method is simple to state. Given training data $D = \{(x_1, y_1)..(x_n, y_n)\}$, the objective is to compute a density over y given a partial testing point x. With this density, one can take the expected value, the most probable value, or whatever. The main idea is to introduce an intermediate variable w, e.g. the weights of a neural network, and then sum it out:

$$p(y|x, D) = \int_w p(y, w|x, D) = \int_w p(y|x, w)p(w|D) \quad (1)$$

$$p(w|D) = \frac{p(w)}{p(D)} \prod_i p(y_i|x_i, w) \quad (2)$$

Note that we assume w is a *separator*: it renders (x, y) conditionally independent from D. There are three common ways to approximate this integral:

MAP Use the single term where $p(w|D)$ is largest (e.g. (Omohundro, 1995)).

Sampling Sample w's from $p(w|D)$ and only perform the integral over the samples (e.g. (Williams and Rasmussen, 1995)).

Analytic Fit a parametric model, e.g. Gaussian, to $p(w|D)$ and perform the integral analytically (e.g. (MacKay, 1992)).

Now suppose $D = \{f_1..f_n\}$ contains IID functions, or more generally point sets, instead of IID points. We want to know about a new function (i.e. point set) g, which has an observed part g^o and missing part g^m. Even if each set f_j contains IID points, we don't necessarily want to simply unite all of the sets since the probability law may vary across sets. Indeed, we may want to imagine there being separate sets of points even if the data wasn't originally presented this way (section 4).

The basic approach is the same. Assume w is a separator for the f_j:

$$p(g^m|g^o, D) = \int_w p(g^m|g^o, w)p(w|D) \quad (3)$$

$$p(w|D) = \frac{p(w)}{p(D)} \prod_j p(f_j|w) \quad (4)$$

The density $p(f_j|w)$ is now a generative model for point sets, e.g. a selector of sinewaves.

2.1 AN EM ALGORITHM

If we further assume that the points in each f_j are independent given a local parameter s_j, we get $p(f_j|w) = \int_{s_j} p(s_j|w) \prod_i p(x_i^{(j)}|s_j)$. Here w plays the role of a prior on s_j, hence learning w from D corresponds to "learning a prior" (Baxter, 1996b) (Omohundro, 1995). It also parallels the hierarchical Bayes approach in (MacKay, 1992).

The combined model can be expressed concisely by a tree-structured Bayesian network with w at the root, s_j as its children, and the points in f_j as the children of each s_j. Many Bayesian network algorithms, including exact algorithms, can be used to efficiently evaluate probabilities on this simple structure (Heckerman, 1996).

Alternatively, we can find the most probable values for w and s_j and use the MAP approximation on all of the integrals. An EM algorithm to find the most probable values is:

E-step For each j, compute s_j which maximizes $p(f_j|s_j)p(s_j|w)$ using the current value of w. Also compute t which maximizes $p(g^o|t)p(t|w)$. t is the local parameter for the testing point set g. This is a MAP approximation to the full expectation over s_j.

M-step Find w to maximize $p(w)p(t|w) \prod_j p(s_j|w)$.

This algorithm was essentially given in (Omohundro, 1995). In that paper, w was a manifold in parameter space and $p(s|w)$ was a Gaussian convolved with the manifold. The E-step fit a linear classifier in this restricted parameter space while the M-step fit a new manifold to the resulting fits.

This algorithm can be used with any hidden variable s_j that renders the points in f_j independent. For example, s_j can be the state sequence in a hidden Markov model, where the algorithm is called Baum-Welch (w is the parameters of the model and each f_j is an observation sequence). As is the case in Baum-Welch and pointed out in (Baxter, 1996b), the dimensionality of w can be much larger than any particular s_j, and this is fine since we may have far more point sets than points in a set.

2.2 GAUSSIAN PROCESSES

The assumption of there being a local separator s_j may not be valid. In this case, a different model for $p(f_j|w)$ must be used. One simple model is a joint Gaussian distribution for all $y(x)$ over the input space x, i.e. f_j is a sample from a Gaussian process (Williams and Rasmussen, 1995). Learning w from multiple instantiations f_j corresponds to fitting a mean and covariance. As pointed out in (Williams and Rasmussen, 1995), this is equivalent to learning the basis functions for spline approximation. However, note that fitting a Gaussian to the y values assumes point-by-point correspondence in the x values across the different f_j.

A useful fact is that if we use the unconstrained maximum-likelihood estimate for the variance, given the functions f_j, then the parameters of the Gaussian density $p(g^m|g^o, w)$ are given by the theory of optimal linear estimation. The theory tells us that the mean y-values in g are a linear combination of the corresponding y-values in f_j (plus an offset for the mean). This means that we can skip the covariance estimation step and fit a linear combination of f_j to g, leading to a very simple and yet mathematically justified algorithm for learning from related tasks. The linear approach of (Tenenbaum and Freeman, 1997) can be understood in this way.

3 LEARNING HOW TO LEARN

Two of the main topics in this literature have been (1) learning a distance metric and (2) learning several tasks with one network. This section shows that they are both variants of the Gaussian process approach.

3.1 LEARNING A DISTANCE METRIC

Suppose we use a nearest-neighbor model for $p(g^m|g^o, w)$, where w is the distance metric:

$$\begin{aligned} p(y|x, g^o, w) &= \delta(y - y_k) && \text{(1 at } y_k, 0 \text{ elsewhere)} \\ k &= \text{argmin}_i w(x, x_i) \\ g^o &= (x_i, y_i) \\ (x, y) &\in g^m \end{aligned} \tag{5}$$

Note that this assumes independence between the points in g. Learning w from related tasks f_j then corresponds to learning a metric. This can be done in several ways, say by feature selection or weighting features (Caruana, 1996).

However, Baxter (Baxter, 1996a) showed that the best metric, over the space of all metrics, is available in closed form given the distribution of functions g. If we only have samples from this distribution (the f_j), then the maximum likelihood estimate of the metric can be computed, without any search, as

$$w(x_1, x_2) = \frac{1}{n} \sum_j \sigma(f_j(x_1), f_j(x_2)) \tag{6}$$

where σ is the loss function. For binary classification (zero-one loss), this "canonical distortion measure" (CDM) reduces to the probability that two samples are classified differently—the same metric used in (Thrun, 1995).

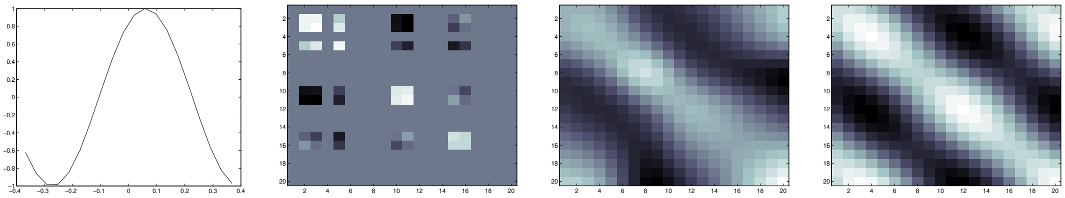
More generally, for either regression (squared-error loss) or classification the CDM is determined by the covariance function $E_j[f_j(x_1)f_j(x_2)]$. Therefore, in either case it suffices to estimate the covariance of the f_j, viewed as samples from a Gaussian process, in order to find the optimal metric for nearest-neighbor on the new task g. Note that nearest-neighbor is insensitive to monotonic transformations of the metric, hence the covariance function. Of course, if the Gaussian process parameters are known exactly, then it is better to use the Gaussian estimator rather than nearest-neighbor.

3.2 MULTITASK LEARNING

Several authors have proposed learning several tasks at the same time using one multilayer perceptron (MLP) (Caruana, 1994) (Baxter, 1995). The tasks constrain each other since they share the same input-to-hidden weights. In this scenario, s_j is the weights of the subnetwork relevant to task f_j. It includes the input-to-hidden and hidden-to-output weights. The prior $p(s_j|w)$ forces all of the s_j to use the same input-to-hidden weights w (Baxter, 1996b).

With linear output units and a Gaussian prior on the hidden-to-output weights, this is easily shown to be a Gaussian process model. For each input x to the network, there is a column vector of hidden unit responses $h(x)$. Collecting the responses to all inputs gives a (possibly infinite) matrix H. Every task f_j is modeled as $o_j{}^{\mathrm{T}} H$ for Gaussian hidden-to-output weights o_j, so the tasks are Gaussian distributed.

Therefore, multitask learning is essentially exploiting the ability of a multilayer perception to model a covariance. A Gaussian process is being fit, but with an



(a) (b) (c) (d)

Figure 1: (a) One of four sinewaves to be learned. (b) Empirical covariance of the training sample. The empty regions correspond to unobserved input locations. (c) Covariance recovered by the MLP. (d) True covariance.

unconventional covariance estimator. Denoting the output of the network for every task on every input by OH, we have that the covariance of the tasks is $(OH)^{\mathrm{T}}(OH) = H^{\mathrm{T}}O^{\mathrm{T}}OH$, therefore the pointwise covariance

$$E_g[g(x_1)g(x_2)] = h(x_1)^{\mathrm{T}}O^{\mathrm{T}}Oh(x_2) \tag{7}$$

for any inputs x_1 and x_2, including those not in the training set. If o_j is standard Gaussian distributed, then $O^{\mathrm{T}}O \approx I$ and the prediction of g^m using the optimal linear combination of the basis H is the same as the conditional mean under a joint Gaussian model with the above covariance. If $O^{\mathrm{T}}O$ is not I, the MLP may be doing worse than the conditional mean since it does not exploit the correlations in the o_j.

The particular covariance which is learned depends on the inductive bias of the MLP and is not in general the same as the unconstrained maximum likelihood solution. This is good since it will interpolate the covariance matrix to unseen input locations, which the unconstrained solution would not. Thus the MLP is a direct competitor with maximum-likelihood techniques for fitting a covariance in the presence of missing data (e.g. (Ghahramani and Jordan, 1994)), which typically assume the covariance is finite. Figure 1 shows an example of using an MLP to fit a covariance.

As discussed in the last section, the optimal distance metric for nearest-neighbor classification and regression is determined by the covariance function. Therefore the MLP approach with (7) can be used, in contrast to the ad hoc method of (Baxter, 1996a) and (Thrun, 1995). The method in those papers models the covariance function directly using a regression network, which is suspect since it (1) minimizes squared error on the elements of the covariance as opposed to the estimate derived from the covariance and (2) doesn't naturally enforce symmetry and positive definiteness.

4 AUTOMATIC PARTITIONING

So far we have assumed that the data was already partitioned into IID sets of points. Suppose we are just given a set of points $D = \{(x_i, y_i)\}$. To apply the preceding algorithms, we need to automatically segment the data into f_j so that each segment can be modeled as an IID sample from a known distribution $p(f_j|w)$, with fixed but unknown w. This is similar to fitting a mixture model, except that a mixture model chooses partitions to make the points within a set IID, not to make the sets themselves IID. An example of this new situation is partitioning a long timeseries into individual sequences whose probability law, e.g. Markov transition matrix, are all chosen from the same underlying distribution (Omohundro, 1995).

The main simplifying assumptions are (1) the number of partitions is known and (2) there exists a local separator s_j which renders the points in f_j independent, as in section 2.1. The idea is to introduce indicator variables z_{ij} which, if all known, would render the partitions independent (Jordan and Jacobs, 1994). When x_i belongs to partition f_j, $z_{ij} = 1$. Applying the MAP approximation to z, s, and w, we want to find the joint maximum of:

$$p(D, z, s, w) = p(w) \prod_j p(s_j|w) \prod_i p(x_i, y_i|s_j)^{z_{ij}} \tag{8}$$

which can be done using a modification of the algorithm from section 2.1:

E-step 1 For each i and j, compute the expected value of z_{ij} which is

$$h_{ij} = \frac{p(x_i, y_i|s_j)}{\sum_j p(x_i, y_i|s_j)} = p(s_j|x_i, y_i) \tag{9}$$

This is the soft assignment of each point to a partition.

E-step 2 For each j, compute s_j to maximize $p(s_j|w) \prod_i p(x_i, y_i|s_j)^{h_{ij}}$ using the current value of w. This is now a weighted maximum likelihood problem where h_{ij} are the weights.

M-step Find w to maximize $p(w) \prod_j p(s_j|w)$. This is the same as before.

This algorithm was suggested in (Omohundro, 1995) and is a generalization of the algorithm for mixture models to the case where the components are coupled by the parameter w. In the case where s_j is the state sequence of HMM j, we get weighted Baum-Welch, where several HMMs compete for data and therefore there may be gaps in the observation sequence for each.

5 TASK CLUSTERING

Finally, we may not want one underlying parameter w which separates all of the tasks f_j, because the tasks are not equally relevant to each other. In this case, we can partition the tasks into groups, where each group has its own w. This is a direct generalization of clustering points to clustering sets of points.

A mixture model could be applied, but a cheaper route is to use a clustering algorithm. The trick is defining a good notion of similarity between point sets f_j. Since the objective is to group f_1 with the f_2 whose generative parameter w is the same, the following (asymmetric) measure seems appropriate:

$$\text{similarity of } f_1 \text{ to } f_2 = \int_w p(f_1|w)p(w|f_2) \tag{10}$$

which can be approximated by $p(f_1|w_{\mathrm{MAP}})$ where w_{MAP} maximizes $p(w|f_2)$. This is essentially the technique used in (Thrun and O'Sullivan, 1996) where w was the distance metric for nearest-neighbor. The algorithm was to compute the optimal metric for task f_2, then test it on task f_1 via cross-validation, using the cross-validation score as a measure of distance to f_2. Yianilos has also studied this similarity measure for nearest-neighbor classification (Yianilos, 1995).

This similarity measure on point sets can be used in other ways. For example, to collect only the tasks relevant to the new task. This is the generalization of "local learning" to sets of points. The task weighting in (Caruana, 1996) could be done this way instead of searching over weights for all tasks.

6 CONCLUSION

Virtually all of the techniques for "learning how to learn" presented in the neural network and machine learning literatures can be understood in terms of a generalization of points to point sets. This observation should cut down on redundant algorithms. It should also add to the interest in these techniques by giving them an intuitive grounding.

An appealing aspect, not often mentioned, is that learning from related tasks offers an elegant way to express prior knowledge: as a series of example tasks. The choice of distance metric for nearest-neighbor, the kernels in a radial basis function network, and the covariance of a Gaussian process can all be understood in this way. The situation is analogous to the use of "virtual examples" to represent conjugate priors in statistics (Heckerman, 1996). Just as one can fit a conjugate prior to an existing prior, one can find the example tasks that underlie the hyperparameter choices made in these learning models. An example is Baxter's proof that using Euclidean distance implies a set of example tasks which are simple hyperplanes (Baxter, 1996a).

References

C. Bregler and S. M. Omohundro, "Surface learning with applications to lipreading," in *NIPS*, pp. 43–50, Morgan Kaufmann Publishers, 1994.

Z. Ghahramani and M. I. Jordan, "Supervised learning from incomplete data via an em approach," in *NIPS*, MIT Press, 1994.

J. Baxter, "A Bayesian/information theoretic model of bias learning," in *COLT*, pp. 77–88, July 1996.

S. Thrun, "Is learning the n-th thing any easier than learning the first?," in *NIPS*, MIT Press, 1996.

R. Caruana, "Learning many related tasks at the same time with backpropagation," in *NIPS*, MIT Press, 1995.

S. M. Omohundro, "Family discovery," in *NIPS*, MIT Press, 1996.

C. K. I. Williams and C. E. Rasmussen, "Gaussian processes for regression," in *NIPS*, MIT Press, 1996.

D. J. C. MacKay, "Bayesian interpolation," *Neural Computation*, vol. 4, no. 3, pp. 415–447, 1992.

D. Heckerman, "A tutorial on learning with Bayesian networks," Tech. Rep. MSR-TR-95-06, Microsoft Research, 1996.

J. B. Tenenbaum and W. T. Freeman, "Separating style and content," in *NIPS*, MIT Press, 1997.

R. Caruana, "Algorithms and applications for multitask learning," in *ICML*, Morgan Kaufmann, 1996.

J. Baxter, "The canonical distortion measure for vector quantization and approximation." `http://keating.anu.edu.au/~jon/research.html`, 1996.

J. Baxter, "Learning internal representations," in *COLT*, ACM Press, 1995.

M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the EM algorithm," *Neural Computation*, vol. 6, pp. 181–214, 1994.

S. Thrun and J. O'Sullivan, "Discovering structure in multiple learning tasks: The TC algorithm," in *ICML*, Morgan Kaufmann, 1996.

P. N. Yianilos, "Metric learning via normal mixtures." `http://www.neci.nj.nec.com/homepages/pny/papers/mlnm/`, 1995.